~~AN APPARATUS AND METHOD FOR COMBINING SIMD RESULTS USING A STALL BASE METHODOLOGY~~

TECHNICAL FIELD

The present invention is generally related to performing SIMD (Single Instruction Multiple Data) instructions and, more particularly, is related to an apparatus and method for performing SIMD instructions (*i.e.,* multiply accumulate operations) using one multiply accumulate (MAC) unit while minimizing the operational latency.

BACKGROUND OF THE INVENTION

SIMD instructions are those instructions that perform the same operation on two or more pieces of data at the same time. A SIMD data word consists of two single precision floating-point numbers, packed into a floating-point word. In an example of a 82-bit floating-point word, the low SIMD data is stored in bits 31-0, and the high SIMD data is stored in bits 63-32. The remaining bits of the example 82 bits (81-64) word are set to a predefined constant.

Currently, two miscellaneous 31(A&B) units and two MAC 41(A&B) units are required to perform SIMD instructions. A miscellaneous unit (MISC) is a unit that performs all operations not requiring multiply accumulate functions, such as all logical functions. A first MAC 41B unit is responsible for the high SIMD word, and the second MAC 41A unit is responsible for the low SIMD word. The MAC 41(A&B) units results are then

combined on a single register file 21. A block diagram of an example of the prior art system architecture to perform SIMD instructions using multiple MAC 41(A&B) units is illustrated in FIG. 1. This prior art implementation, which indicates two functional units, includes two MISC 31(A&B) full precision MAC units (low 41A& high 41B), and two single precision SIMD MAC units 11(A&B). It can simultaneously perform any of two SIMD instructions, one SIMD and one non-SIMD instruction, or two non-SIMD instructions.

Thus, a heretofore unaddressed need exists in the industry to be able to perform SIMD instructions using a single MAC unit while minimizing the operational latency.

SUMMARY OF THE INVENTION

The present invention provides an apparatus and method for performing SIMD instructions (*i.e.,* multiply accumulate operations) using one MAC unit while minimizing the operational latency.

Briefly described, in architecture, the apparatus can be implemented as follows. A MAC unit generates a first half of a data result in a first time period and a second half of a data result in a later time period. A deferred register stores the first half of a data result while the second half of the data result is being computed. A MISC logic is used to determine when to release the first half of a data result stored in the deferred register in order to synchronize the first half of a data result with the second half of said the result.

The present invention can also be viewed as a method for performing SIMD instructions using one MAC unit while minimizing the operational latency. In this regard, the method can be broadly summarized by the following steps: A MAC unit generates a first operand data result. The first operand data result is input into a deferred register. The MAC unit generates a second operand data result. While the MAC unit generates a first and second operand data result, an exception result is generated by a MISC logic. The MISC logic places the first operand data result and said second operand data result into a buffer if the MISC logic determines that the first operand data result and the second operand data result are valid. The MISC logic places the exception result into the buffer if the MISC logic determines that the first operand data result and the second operand data result are invalid.

Other features and advantages of the present invention will become apparent to one with skill in the art upon examination of the following drawings and detailed description. It is intended that all such additional features and advantages be included herein within the scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the

present invention. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

FIG. 1 is a block diagram of a prior art system architecture to perform SIMD instructions using multiple MAC units.

FIG. 2 is a block diagram of a possible example of the system architecture to perform SIMD instructions using a single MAC unit of the present invention.

FIG. 3 is a flow chart of a possible example for how data moves through the pipeline of the system architecture to perform SIMD instructions using a single MAC unit as shown in FIG. 2.

FIG. 4 is a timing diagram of a possible example for how data moves through the pipeline of a possible example of the system architecture to perform SIMD instructions using a single MAC unit as shown in FIG. 3.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Reference will now be made in detail to the description of the invention as illustrated in the drawings. While the invention will be described in connection with these drawings, there is no intent to limit it to the embodiment or embodiments disclosed therein. On the contrary, the intent is to cover all alternatives, modifications, and equivalents included within the spirit and scope of the invention as defined by the appended claims.

Illustrated in FIG. 2 is a possible example of the system architecture to perform SIMD instructions using a deferred register and a single MAC

unit of the present invention. As shown, the register file 21 provides operand (A-C) data on operand busses A 22, B 23 and C 24. Operand busses A-C (22-24) transfer operand data from the register file 21 to logic 32 in the MISC 31, and to the MAC 41. There is no logic in MISC 31 between register file 21 and MAC 41 for the operands. The MAC 41 receives the operand (A-C) data on operand busses A-C (22-24), in logic 42. Logic 42 also receives control opcodes from the external control unit that are necessary to compute the appropriate result.

Using the operand (A-C) data and control opcodes, MISC logic 32 generates a series of four sets of signals and their complements to control the various bus drivers in both the MISC 31 and MAC 41. The four sets of signals include:

A = miscop + macop * misc_result + simdop. Generating signal A will enable the data bus 36 to transmit data to the result data bus 71A. The data bus 36 is able to transmit data to the result data bus 71A when signal 72 enables buffer/driver 33.

B = miscop + macop * !simd * miscresult + macop * misc_result_high * simdhigh. Generating signal B will enable the data bus 37 to transmit data to the high half result data bus 61A. The data bus 37 is able to transmit data to the high half result data bus 61A when signal 62 enables buffer/driver 34. The high half result data bus 61A transmits data to deferred register 80 for storage. The deferred register 80 stores the first half of a data result while the second half of the data result is being computed. A MISC logic is used to determine when to release the first half of a data

result stored in deferred register 80 in order to synchronize the first half of a data result with the second half of the result.

C = miscop + macop * !simd * miscresult + macop * misc_result_low * simdhigh. Generating signal C will enable the data bus 38 to transmit data to the result data bus 51A. The data bus 38 is able to transmit data to the result data bus 51A when signal 52 enables buffer/driver 35.

D = macop * !misc_result_low * simdhigh. Generating signal D will enable the data bus 61B to transmit data from deferred register 80 to the result data bus 51B. The data bus 61B is able to transmit data to the result data bus 51B when signal 75 enables buffer/driver 27. These signals arrive during time x and time y on the timing diagram (FIG. 4). Basically, these signals can be considered FP4 signals that might be qualified by the simdhigh/low signals.

In SIMD mode, result control A is valid during time y, result control B is valid for both time x and y, and result control D&C are valid for time y. In non-SIMD mode, these signals are valid in FP4.

The four sets of signals listed above are comprised of the following instructions and signals:

miscop = there is an instruction for the MISC unit;

macop = there is an instruction for the MAC unit;

misc_result = a non-SIMD MAC instruction has results generated by the MISC unit;

misc_result_low = a SIMD MAC instruction has low half data generated by the MISC unit;

misc_result_high = a SIMD MAC instruction has high half data generated by the MISC unit;

simdhigh = asserted during cycle (the 2nd FP4) in which the results for the high half SIMD are generated (It is assumed that the signal simdhigh is only active if the signal simd is active);

simd = there is a SIMD instruction for either MISC or MAC.

These signals are generated by the MISC 31, based upon the opcodes and operands. The operands are received by the MISC 31 from the register file 21. The opcodes come from an external control unit (FPU Control) (not shown) that communicates with the main instruction fetch unit. The FPU Control and MISC 31 units are responsible for the correct staging of pipelined control information.

The bus drivers in FIG. 2 will drive only when their select line is asserted. An example will be illustrated with regard to Fig. 3 herein defined in detail below. Note that this diagram (FIG. 2) demonstrates only one functional unit. It is contemplated by the inventors that there can be two functional units, one on either side of the register file 21, similar to FIG. 1. The configuration of the present invention (FIG. 2) is able to simultaneously perform two SIMD instructions (4 total operations), one SIMD and one non-SIMD instruction, or two non-SIMD instructions.

While the implementation of the present invention (FIG. 2) requires an additional cycle of latency, that of FIG. 1, the system architecture of the present invention permits a SIMD instruction to be executed in parallel with another instruction (SIMD or non-SIMD). Of course, it is contemplated by

the inventors that other methods of performing the bus multiplexing are possible, such as repeating multiplexers, etc.

Illustrated in FIG. 3 is a flow chart of a possible example for how data moves through the pipeline of a possible example of the system architecture to perform SIMD instructions using a single MAC unit as shown in FIG. 2. First, the register file 21 (FIG. 2) drives data on the operand busses A 22, B 23 and C 24, for two consecutive cycles.

On the first cycle at step 101, the MAC 41 latches the low operand data into its high data latches in logic 42 and begins the operations on the next cycle. The opcode information also arrives from the external control unit at step 101.

On the second cycle at step 102, the MAC 41 will start operation on the low operand data and latch the high operand data into the high data latches of logic 42. The MISC 31 will latch both high and low operand data at step 102, and opcodes arrive via busses (22-24), again, from the external control unit.

On the third cycle, at step 103, the MAC 41 continues operation on the low operand data and starts operation on the high operand data. The MISC 31 begins its operation on both the high and low operand data at step 103. A second instruction (either SIMD or non-SIMD) may have its operands/opcodes delivered to the MISC 31 and MAC 41 units to start on the next cycle.

On the fourth cycle, at step 104, the MAC 41 continues operation on both the lower and higher operand data. A third instruction can also enter

the busses (22-24) on this cycle. This is a fully pipelined system and once the instructions leave a certain clock state (*e.g.*, FP1, FP2, FP3, FP4, WRB) another instruction, SIMD or non-SIMD can enter that clock stage.

On the fifth cycle, at step 105, the MAC 41 delivers the low operand data result onto the high half result data bus 47. The low operand data result is then transmitted to the high half result data bus 61A. This is accomplished by a using signal 62 from logic 32 in MISC 31 as input into inverter 63 to generate enable signal 64. This enable signal 64 authorizes buffer/driver 44 to transmit the lower operand data from the high half result data bus 47 to the high half result data bus 61A. Signal 62 is also input in its original value into buffer/driver 34. This original value for signal 62 disables buffer/driver 34 from transmitting the operand data result from logic 32 in MISC 31 onto high half result data bus 61A. The low operand data result from the high half result data bus 61A is latched into the deferred register 80. Also in the fifth cycle, the MAC 41 continues operation on the high operand data.

On the sixth cycle, at step 106, the MISC 31 will indicate whether to use the MAC 41 results or the MISC 31 exceptional results. The MISC 31 indicates which results are to be utilized by generating the signals on signal lines 52, 62, 72 and 75, respectively. These signals cause the appropriate bus drivers 25-27, 33 - 35, 43-45, 53, 63 and 73 to place result data on the appropriate result bus (51A, 61A or 71A).

In the example where the MISC 31 indicates that the MAC 41 results are to be utilized, the MISC 31 causes the appropriate bus drivers 25-27,

33-35, 43-45, 53, 63 and 73 to place result data on the appropriate result bus, respectively. The MISC 31 generates the following signals, illustrated in the table below, on signal lines 52, 62, 72 and 75, respectively, to cause the appropriate bus drivers to place result data on the appropriate result bus (51A, 61A or 71A).

In the example where the MISC 31 indicates that the exceptional results be utilized, the MISC 31 causes the appropriate bus drivers 25-27, 33-35, 43-45, 53, 63 and 73 to place result data on the appropriate result bus (51A, 51B, 61A or 71A). The MISC 31 generates the following signals, illustrated in the table below, on signal lines 52, 62, 72 and 75, respectively, to cause the appropriate bus drivers to place result data on the appropriate result bus.

The cases are these:

CASE	EXCEPTIONS	BUFFER/DRIVERS ON	SIGNALS ACTIVE
Case 1	No exceptions	27, 44, 33	75, 64, 72
Case 2	Low exception	35, 26, 44, 33	52 76 64, 72
Case 3	High exception	27, 34, 33	75, 62, 72
Case 4	Both exceptions	35, 26, 34, 33	52, 76, 62, 72
Case 5	Non Simd, no exception	45, 26, 44, 43	54, 76, 64, 74
Case 6	Non Simd, exception	35, 26, 34, 33	52, 76, 52, 72
Case 7	MISCOP	35, 26, 34, 33	52, 76, 62, 72

If the MISC 31 does not detect an exceptional case for the high mantissa, the MAC 41 delivers the high operand data result onto the high half result data bus 61A. If the MISC 31 does not detect an exceptional case for the low mantissa, the deferred register 80 drives be lower operand data result onto the lower half result data bus 51A.

Whenever the MISC 31 detects an exceptional case, it will be responsible for delivering the result. In any of the SIMD cases, the MISC 31 will deliver the exponent result. The MISC 31 will deliver the exponent result from buffer/driver 33 by generating a signal on signal lines 72.

On the seventh cycle, at step 107, the combined result is then written to the register file 21 (FIG. 2).

While the example of the system architecture of the present invention utilizes a four cycle latency, it is contemplated by the inventors that other cycle latencies may be utilized.

Illustrated in FIG. 4 is a timing diagram of a possible example of how data moves through the pipeline of a possible example of the system architecture to perform SIMD instructions using a single MAC 41 as described above with regard to FIG. 3.

First, the register file 21 (FIG. 2) drives low operand data 114 and high 115 on the operand busses A 22, B 23 and C24 (FIG. 2), for two consecutive cycles as shown by signals 131 and 132.

On the first cycle, the MAC 41(FIG. 2) latches the low operand data 114 into its high data latches and begins the operations on the next cycle. The arrival of the low operand data is shown by signal 131.

On the second cycle, the MAC 41 (FIG. 2) will start operation on the low operand data (114 at FP1) and latch the high operand data 115 into the high data latches. The arrival of the high operand data is shown by signal 132.

On the third cycle, the MAC 41 (FIG. 2) continues operation on the low operand data (114 at FP2) and starts operation on the high operand data (115 at FP1). The MISC 31 (FIG. 2) begins its operation on both the high and low operand data (114 at FP2 and 115 at FP1).

On the fourth cycle, the MAC 41 continues operation on both the lower and higher operand data (114 at FP3 and 115 at FP2). This is a fully pipelined system and once the instructions leave a certain clock state (e.g., FP1, FP2, FP3, FP4, WRB) another instruction, SIMD or non SIMD can enter that clock stage. The MISC 31 generates signals 52, 62 and 72, which control the output buffer/drivers 43, 44 and 45 (FIG. 2). These output buffer/driver 43, 44 and 45, control the data that gets put on result data busses 51A, 61A and 71A (FIG. 2) in the fifth cycle.

On the fifth cycle, the MAC 41 delivers the low operand data result 133 onto the high half result data bus 47(FIG. 2). Signals on data busses 46, 47 and 48 (FIG. 2), are generated by the MAC 41 are dependent upon the operand and opcodes input on operand busses 22-24. For SIMD instruction, high half result data bus 47 is the most significant output bus. For non-SIMD instruction, all three busses (*i.e.*, 46, 47 and 48 (FIG. 2)) are significant output busses. The low operand data result 133 is then transmitted to the high half result data bus 61A. The low operand data

result 133 from the high half result data bus 61A is latched into the deferred register 80 (FIG. 2). Also, in the fifth cycle, the MAC 41 (FIG. 2) continues operation on the high operand data (115 at FP3).

On the sixth cycle, the MAC 41 (FIG. 2) delivers the high operand data result 134 onto the high half result data bus 47. The MISC 31 will indicate whether to use the MAC 41 results or the MISC 31 exceptional results.

On the seventh cycle, the combined result is then written to the register file 21 (FIG. 2)

It should be emphasized that the above-described embodiments of the present invention, particularly, any "preferred" embodiments, are merely possible examples of implementations, merely set forth for a clear understanding of the principles of the invention. Many variations and modifications may be made to the above-described embodiment(s) of the invention without departing substantially from the principles of the invention. All such modifications and variations are intended to be included herein within the scope of the present invention and protected by the following claims.